基于 Spark 的改进 K-means 算法的并行实现 *

杜佳颖，段隆振，段文影，卜秋瑾

(南昌大学 信息工程学院，南昌 330031)

摘　要：针对 K-means 聚类算法存在的不足，提出了改进 K-means 来提高算法的性能，利用简化后的轮廓系数作为评估标准衡量 K-means 算法中 *k* 值，采用 K-means++完成 K-means 算法初始中心点的选择。设置好 *k* 值以及初始中心点后使用形态学相似距离作为相似度测量标准将数据点归属到距离最近的中心点形成的簇中，最后计算平均轮廓系数确定合适的 *k* 值，并在 Spark 上实现算法并行化。通过对四个标准数据集在准确性，运行时间和加速比三个方面的实验表明，改进后的 K-means 算法相对于传统的 K-means 算法和 SKDK-means 算法不仅提高了聚类划分质量，缩短了计算时间，而且在多节点的集群环境下表现出良好的并行性能。实验结果分析出提出的改进算法能有效提高算法执行效率和并行计算能力。

关键词：聚类算法；简化轮廓系数；形态学相似距离；相似性度量

中图分类号：TP301.6　　doi: 10.19734/j.issn.1001-3695.2018.07.0526

Parallel implementation of improved K-means algorithm based on Spark

Du Jiaying, Duan Longzhen, Duan Wenying, Bu Qiujin

(College *of Information Engineering, Nanchang University, Nanchang* 330031, *China*)

Abstract: Aiming at the deficiency of K-means clustering algorithm, this paper proposes an improved algorithm with the use of simplified silhouette coefficient as the evaluation criterion to measure the *k* value in K-means to boost the algorithm performance. The K-means++ algorithm is used to choose the initial center points in the K-means algorithm. After setting the *k* value and the initial center point, morphology similarity distance is used as the similarity measurement standard to assign the data points to the cluster formed by the closest center point. And finally calculate the average silhouette coefficient to determine the appropriate *k* value. The improved algorithm in this paper is implemented on Spark. Experiments on accuracy, run-time and speedup of four standard datasets show that the improved K-means algorithm not only improves the quality of clustering division compared with the traditional K-means algorithm and SKDK-means algorithm, but also shortens the calculation time, showing good parallel performance in a multi-node cluster environment. The experimental results suggest that the improved algorithm proposed in this paper can effectively improve the algorithm execution efficiency and parallel computing ability.

Key words: clustering algorithm; simplified silhouette coefficient; msd distance; similarity measurement

0　引言

聚类分析是数据挖掘领域一项重要的研究课题，它是探索数据挖掘的主要任务，在统计分析数据方面它被广泛使用在多个领域，包括图形图像处理[1]、信息检索[2]、文本挖掘[3]等方面。K-means 算法是聚类分析中广泛使用的算法，它存在两点不足：*k* 值的不确定导致每次迭代后的实际结果与理想值有偏差[4]；其次，初始点的随意选择使结果不稳定，使得最终的聚类结果将产生很大的误差值。

随着当今数据的急剧上升，一些并行计算模型包括 MapReduce 和 Spark 开始流行起来，由于 MapReduce 的中间计算结果保存在硬盘上，而迭代计算需要在 Hadoop 分布式文件系统(HDFS)重复地进行读写操作导致较高的 I/O 负荷和时间消耗，所以 MapReduce 框架不适合迭代计算[5]。然而，Spark 能克服 Hadoop 存在的不足，它作为一种基于内存的分布式框架可以把中间计算结果放于内存中，而且启动速度很快。

在当今大数据的时代背景下，许多专家学者相继提出了许多基于 Spark 的改进算法。徐鹏程等人[6]利用 Canopy 算法确定簇类数并使用最大最小距离选取初始点，而 Canopy 算法需要预先确定好阈值 T1 和 T2,结果的好坏很大程序得受阈值的影响。邓青等人提出基于最小准则函数的 K-means 改进算法，通过 i 次取样每次得到一个聚类中心合并后确定 *k* 个初始中心[7]。宋董飞等人[8]提出了并行算法，该算法使用 kd-tree 选择初始点并利用它的最近邻搜索减少距离计算过程，加快聚类速度使其有效应用于海量数据中。后两位学者主要针对初始点的选择作出了改进，而 k 值是不确定的，选择合适的 *k* 值对于聚类结果准确率有着必要的影响。因此，本文采用简化轮廓法作为有效性评估方法确定簇类个数 *k* 值。对于 K-means 初始点随机选择的问题，Arthur 等人提出了 K-means++算法[9]，该算法可以自动获取到 K 个初始点，避免随机选择初始点带来的不稳定性，同时也能加快算法收敛速度，因此本文引入 K-means++算法来提高算法性能。

利用上述提到的概念，为了加快算法聚合速度，提高算

收稿日期：2018-07-20；**修回日期：**2018-08-28　**基金项目：**国家自然科学基金资助项目（61070139，81460769）

作者简介：杜佳颖（1995-），女，江西景德镇人，硕士研究生，主要研究方向为数据挖掘（980639652@qq.com）；段隆振（1961-），男，江西九江人，教授，博导，主要研究方向为数据挖掘、管理信息系统、决策支持系统等；段文影（1989-），男，江西九江人，博士研究生，主要研究方向为人工智能；卜秋瑾（1994-），女，河南周口人，硕士研究生，主要研究方向为数据挖掘.

法准确率，并能在数据量较大的情况下通过并行计算获得较快的处理时间，得到较优的聚类结果。本文提出基于 Spark 的 K-means 算法改进及并行化实现，并用标准 UCI 数据集说明改进后算法的性能有所提升。

1　相关工作

1.1　Spark 框架

Apache Spark 是一个开源的大数据聚类计算的分布式框架，它提供了数据并行性以及高容错性等良好的特性。它还提供了 Python，Java 和 Scala 的简单的编程接口[10]。Spark 抽象出来的弹性分布式数据集(RDD)使得 Spark 能更好得处理迭代任务，RDDs 在数据集群中只可读，存储方式有两种，一是以文件的形式保存在外界存储系统中如 HDFS，二是通过其他的 RDDs 产生。RDDs 操作形式有两种分别是 transformations 和 actions，前者指的是从已有的数据集中创造新的数据集出来如 RDDs，后者将数据集上计算得到的结果返回给 Driver 程序如 HDFS。如下图 1 所示。

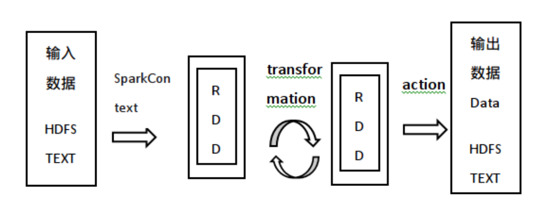


图 1　spark 编程模型

Fig. 1　Spark programming module

Spark 的集群模式是以 master-slave 的形式存在的，由一个 Master 节点和多个 Slave 节点组成，其中 master 节点负责整个集群任务的分配，调度和监测工作，而 slave 节点执行 worker 任务，它主要负责计算和产生状态报告。Spark 支持多种形式的运行模式包括单机方式，集群方式，在 yarn 上和在 Mesos 上等多种方式。

1.2　K-means++算法

K-means++算法是一个用来选择 K-means 初始中心点的聚类算法，在 2007 年首次被 David Arthur 和 Sergei Vassilviskii 等人提出[9]，它避免了传统 K-means 算法随机选择初始点带来的不足，也是一种改进算法，算法如下图 2 所示。

```
Algorithm 1 k-means++(k) initialiazation
C ←sample a point uniformly at random from X
while |C| < k do
    Sample x∈X with probability d²(x,C)/φX(C)
    C←C∪{x}
end while
```

图 2　K-means++算法流程

Fig. 2　K-means algorithm flow chart

具体的算法流程如下所示：

a)从数据集中随机选择一点作为首个中心点；

b)对于数据集中的每一个样本点 x 分别计算它与当前已有质心之间的最短距离，记为 D(x)；

c)选择一个新的点作为新的中心点：根据概率公式，D(x)越大的点作为中心点的概率越大；

d)重复步骤 b) c)直到选择出 k 个中心点；

定义 1　简化轮廓法[11]。

$$ss(i)=\frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \tag{1}$$

其中：a(i) 和 ,b(i) 分别用以下公式表示：

$$a(i)=d_E(x_i,c_h) \tag{2}$$

$$b(i)=\min d_E(x_i,c_l);(lh) \tag{3}$$

其中：$d_E(x_i,c_h)$ 表示点 i 到所在的簇的簇中心的欧氏距离，$\min d_E(x_i,c_l)$ 表示点 i 到其他簇中心的距离的最小值，与原始轮廓法不同的是简化轮廓法采用的是数据点到簇中心的距离，而原始轮廓法则是数据点到所有点的平均距离。它相对于原始轮廓法能加快计算时间，在不同的 K 值下得到更好的结果。

数据集 D 的平均简化轮廓值 ss 为

$$ss=\frac{1}{n}\sum_{i=1}^{n}ss(i) \tag{4}$$

其中：ss 是数据集 D 所有点的平均轮廓值，数值范围在-1 到 1 之间，-1 表示簇类划分效果不好，1 表示效果很好，因此应取接近 1 的 k 值。

定义 2　明可夫斯基距离：

$$D(j,k)=\left(\sum_{i=1}^{m}\left|X_{ji}-Y_{ki}\right|^{p}\right)^{\frac{1}{p}} \tag{5}$$

其中：m 是向量维度，X_{ji} 是向量 j 的第 i 个属性值，Y_{ki} 是向量 k 的第 i 个属性值，$|X_{ji}-Y_{ki}|$ 表示的是点与点在属性 i 上的距离。当 p =1 时表示曼哈顿距离，p =2 时表示欧氏距离。

定义 3　形态学相似距离 MSD(morphology similarity distance) [12]。

$$d_{MSD}=ED\times(2-ASD/SAD) \tag{6}$$

$$d_{ASD}=\left|\sum_{i=1}^{m}(X_{ji}-Y_{ki})\right| \tag{7}$$

式(7)中 ASD 代表的是两个向量属性差值之和的绝对值，X_{ji} 是第 j 个向量的第 i 个属性值，Y_{ki} 是第 k 个向量的第 i 个属性值，m 为特征维度。如果 X_{ji} 都比 Y_{ki} 大或都比 Y_{ki} 小，ASD 距离就变成曼哈顿距离了。ED 表示欧氏距离，SAD 是曼哈顿距离。表 1 展示了向量间的距离计算例子。

表 1　向量间的距离计算

Table 1　Different distances between two vectors

特征差	ASD	SAD	ED	MSD
(-1,0,2)	1	3	2.236	3.727

可以看出，如果 ASD/SAD=1 即 ASD 距离等于曼哈顿距离时，MSD 距离就是欧式距离。

2　改进 K-means 算法及在 Spark 上的并行化

2.1　改进 K-means 算法

本文提出的改进 K-means 聚类算法的目的是提高算法的准确性和加快聚类收敛速度，其中心思想是：首先预设一个 *k* 值，通过 K-means++算法得到 *k* 个初始中心点，在不断迭代的过程中，利用 MSD 距离将所有的数据点划分到距离最近的簇中，然后重新计算各个簇中所有数据点的中心点作为新的簇中心，重复此过程直到簇中心不再变化，最后计算不同 *k* 值下聚类对应的简化轮廓系数，最终确定合适的 *k* 值以及对应的簇类结果。

2.2　基于 Spark 的 K-means 改进算法的并行实现

为了能有效地提高算法的并行计算能力，本文提出了基于 Spark 的改进 K-means 算法，主要是围绕“map”和“reduce”两个操作进行的，算法流程如图 3 所示。

并行化过程中每一个分区都对应一对 map 操作和

Reduce 操作,并且同时进行互不影响。mapPartition 算子将每块分区的数据点分配给最近的簇中心，每个 map 操作并行完成。ReduceByKey 算子将所有分区块中的同属一个簇中心的数据点融合，并更新簇中心，该 reduce 过程同样并行地进行。每次迭代都会同时进行多次 map-reduce 操作，对比不同 *k* 值下的简化轮廓系数确定最终的簇类结果。

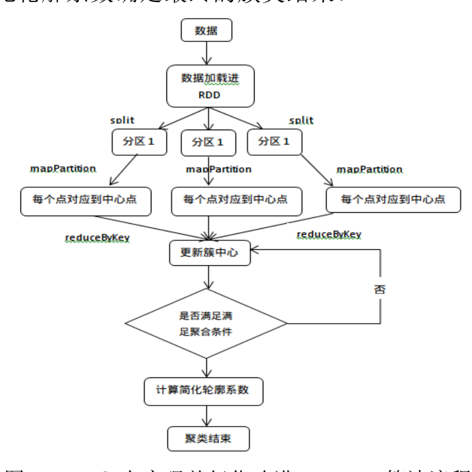


图 3　Spark 上实现并行化改进 K-means 算法流程

Fig. 3　Improved K-means implementation flow chart based on Spark

图 4 展示了基于 Spark 的改进 K-means 算法，大致分为数据分区，聚合和验证三个阶段。

1)数据分区阶段

a)准备 Spark 环境它可以在内存中完成计算将中间结果储存在内存中无须输出到硬盘;b)从 HDFS 读入数据进内存里，加载成 RDD 的形式;c)dense 将数据转换为密集型数据类型，通过 split 操作将数据分片然后转换为<key,value> 键值对，将这些分片后的数据分配到不同的计算节点中，最终通过 mapPartition 算子将每个节点中的数据片划分给不同的簇中心。Cache 使得数据能够在内存中持久化，而每一块分区都是一个 RDD。

2)聚合阶段

该阶段包括步骤 4~8，*k* 值是由人为合理设置的，采用 K-means++选取 *k* 个初始化中心点，将这些选取好的点 broadcast 给各个分区，再通过本文提出来的相似性度量方法 MSD 距离把数据点分到距离最近的中心点形成的簇中。然后依据 reduceByKey 算子把<centerId,pointList>中 centerId 相同的数据点聚集在一起形成新的分区，统计每个簇中数据点的个数，并重新更新簇中心。如果当前迭代次数小于最大迭代次数而且中心点还未收敛，则继续计算新的簇中心，否则结束迭代完成聚合过程。

3) 验证阶段

步骤 9 计算整个数据集的平均简化轮廓值，用它作为 K-means 聚类算法有效性的评估测量标准选择合适的 *k* 值。

```
输入:数据样本 data，聚类数 k，最大迭代次数 maxIteration，
初始中心模式 initialization，随机种子 seed

(1) val sc = new SparkContext(conf);
(2) val data = sc.textFile(path); //加载数据
(3) val parsedData = data.map(s => Vectors.dense(s.split(' ').map(_.toDouble))).cache(); //将数据转化为
RDD[Vector]的格式并持久化
(4)LocalKMeans.kMeansPlusPlus(r, myCenters, myWeights, k, 30); //用 k-means++获取 k 个新的中心点
(5) val bcNewCenters  = data.context.broadcast(newCenters); //将中心点广播给 worker 节点
(6) points.foreach{point=>val(bestCenter, cost)=findClosestWithMSD(thisCenters, point)}//msd 距离找到距离最
近的中心点
(7) counts.indices.filter.iterator{reduceByKey{case ((sum1, count1),(sum2, count2))=>(sum1,count1
+count2);}collectAsMap().mapValues {case (sum,count)=>distanceMeasureInstance.centroid(sum, count)} //通过
reduceByKey 统计各个簇包含的数据点
(8) distanceMeasureInstance.isCenterConverged(centers(j), newCenter,  epsilon);//判断中心点是否收敛
(9) computeSimplifiedSilhouetteScore(dataset,clusterIds,pointClusterId);//计算所有样本的平均轮廓值确定 k 值
```

图 4　Spark 上实现并行化改进 K-means 算法

Fig. 4　Improved K-means algorithm based on Spark

3　实验结果分析

本文在 Spark 分布式集群上实现了改进 K-means 算法，它由一台 master 主节点和三台 slave 从节点组成，每台机器的硬件配置都是：CPU 双核 2.5 GHZ，内存 4 GB，硬盘 500 GB。操作系统是 64 位 Ubuntu16.04。测试采用的数据集是从 UCI 标准数据库下载的 Iris，Wine，Glass，Flame 四个数据集,如表 2 所示。

表 2　所用数据集

Table 2　Datasets of used

数据集	记录数	维度	聚类数
Iris	150	4	3
Wine	178	13	3
Glass	214	9	7
Flame	240	2	2

a)准确率。由于数据集每条记录都有对应的分类项，准确率指的是簇划分正确的的点的个数与样本点总个数的比值用来反映算法的准确程度。图 5 比较了单机环境下改进后的 K-means 算法与传统的 K-means 算法和文献[8]提出的 K-means 算法在四个不同的数据集下准确率的对比，从图中可以看出，每个数据集下改进后的算法比另外两个算法的准确率都有显著提升，原因在于改进算法使用简化轮廓系数更好地确定 *k* 值，利用 K-means++得到 *k* 个中心点并通过 MSD 距离有效将数据点划分到簇中提升算法准确度。

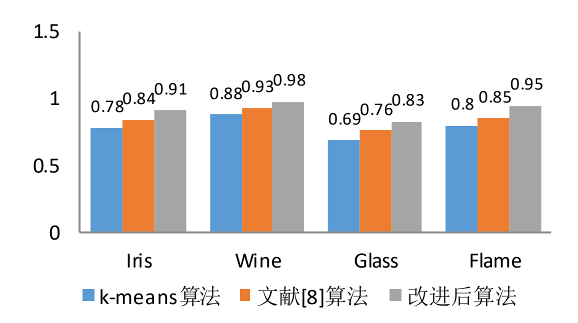


图 5　改进算法与其他算法的准确率比较

Fig. 5　Accuracy of improved K-means compared with other algorithms

b)运行时间。运行时间用来评估改进算法是否提高了算法的有效性，它决定了运行速度。图 6 比较了这三种算法在不同数据集下运行时间。可以看出并行后的改进算法在时间消耗上有一定的减少，因为该算法减少了迭代次数加快收敛速度从而有效提高了算法的计算性能。而在并行计算过程中，

比较耗时的阶段是当利用 MSD 距离计算各个簇中所有数据点到中心点的最短距离时，随着数据集增大从而计算量增大，运行时间因而随之变长。图 7 比较了改进后的算法在并行环境下不同个数的节点及不同类型数据集下的运行时间，可以发现，随着节点增加，算法运行时间呈下降趋势且逐渐趋于平缓，原因在于节点间通信时间增加抵消了集群带来的增益。

c)加速比。为了更加准确地测量并行算法的有效性，加速比用来衡量程序并行化的性能，它指的是同一个任务在串行计算和并行计算系统中运行消耗的时间比率。计算公式为 R=Ts/Tp，其中 Ts 是以传统的串行方式运行算法消耗的时间，Tp 是算法在由 s 个节点构成的集群环境中计算的时间。R 越大表示算法在并行环境中所需要的计算时间越少，从而说明算法的并行性越高。为了测试改进后的算法的并行能力，实验同样基于四个数据集进行对比，改进的算法的效果如图 8 所示。可以看出本文提出的改进算法随着节点数增加，加速比也随之增大。纵向对比，在具有同样多节点的环境下，较大的数据集具有更高的加速比。当计算节点增多时，每个节点需要计算的数据集会有所下降，从而并行计算时间 Tp 变小，即加速比增大。同理，当节点数保持不变，随着数据集增大串行计算时间 Ts 随之增大，因此并行性也同样会提高。然而，随着节点数的增多，加速比上升的速度有所下降。原因在于该实验使用的数据不够大，数据加载时正好能装进单个节点的内存中，当节点变多，它们间的存在的通信开销也会增大，在一定程度上抵消了节点增加带来的性能提升。

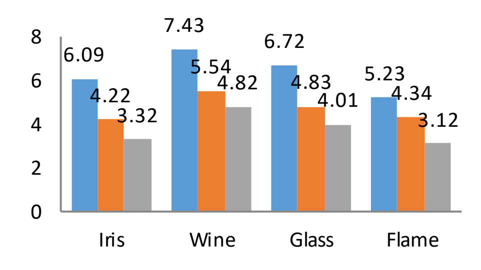


图 6　基于 Spark 改进算法与其他算法运行时间的比较

Fig. 6　Running time of improved K-means compared with other algorithms

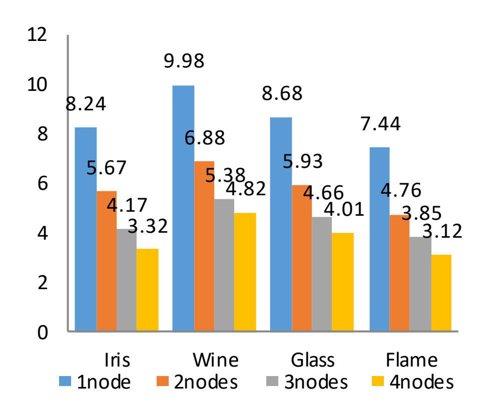


图 7　基于 Spark 改进算法在不同节点下运行时间的比较

Fig. 7　Running time of improved K-means among different number of nodes based on Spark

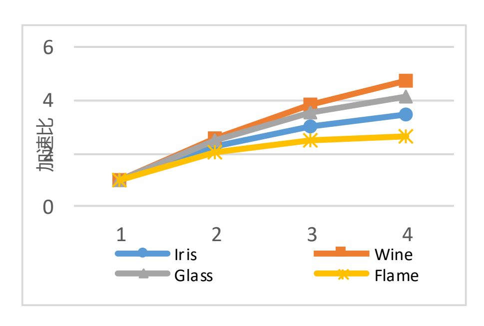


图 8　基于 Spark 的改进 K-means 算法的加速比

Fig. 8　Speedup rate of improved K-means on Spark

综上所述，本文将提出来的改进 K-means 算法和文献[8]中的算法以及传统 K-means 算法分别从运行时间，准确率两个方面加以对比分析，改进后的算法在运行时间上和准确率上都有明显的性能提升。从加速比角度分析出该算法具有良好的并行计算能力。因此总体看本文改进算法优于未改进的算法。

4　结束语

本文提出了改进后的 K-means 算法及基于 Spark 分布式计算框架的并行实现。该算法利用 K-means++算法选择出 k 个初始中心点并以 MSD 距离作为相似性度量距离划分数据点到相应的簇中，然后根据简化轮廓系数法确定合适的 k 值作为簇中心个数。实验结果表明本文提出的改进算法能有效提高算法准确率，减少运行时间，并通过在 Spark 集群环境中比较多个节点条件下的并行计算性能从而说明改进后的算法在并行环境中能发挥良好优势。本文搭建的是小型集群环境，用以处理小规模的数据量。在未来的研究中，笔者计划扩大集群规模，继续优化算法增强算法的并行性，使用更大数据级别的数据检验算法的鲁棒性。

参考文献：

[1] 李斌，李蓉，周蕾. 分布式 K-means 聚类算法研究与实现 [J]. 软件，2018，39 (1):35. (Li Bin, Li Rong, Zhou Lei. Research and implementation of distributed k-means clustering algorithm [J]. Computer Engineering & Software, 2018, 39 (1): 35.)

[2] Mahbub U, Imtiaz H, Ahad M A R. Action recognition based on statistical analysis from clustered flow vectors [J].Signal, Image and Video Processing, 2014, 8(2): 243–253.

[3] Lipka N, Stein B, Anderka M. Cluster-based one-classensemble for classification problems in information retrieval [J]. SIGIR, 2012, 1041-1042.

[4] Joshi　K D, Nalwade P S. Modified K-means for better initial centers. International Journal of Computer Science and Mobile Computing, 2013, 2 (7): 219-223.

[5] Srirama S N, Batrashev O, Jakovits P, et al, Scalability of parallel scientific applications on the cloud [J]. Scientific Programming, 2011, 19(2): 91–105.

[6] 徐鹏程，王诚. K-means 算法改进及基于 Spark 计算模型的实现 [J]. 南京邮电大学学报：自然科学版,2017,37(4):113-118. (Xu Pengcheng, Wang Cheng. Improvement of K-means algorithm and implementation based on Spark computing model [J]. Journal of Nanjing University of Posts and Telecommunications:Natural Science Edition,2017, 37(4): 113-118.)

[7] 邓青，杨宁. 基于 Spark 框架的改进并行 K-means 算法研究 [J]. 智

能计算机与应用. 2018, 8(1): 76-78. (Deng Qing, Yang Ning. Research of improved parallel K-means algorithm based on Spark framework [J]. Intellij Computer and Applications. 2018, 8(1): 76-78.)

[8] 宋董飞, 徐华. 基于 Spark 的 K-means 改进算法的并行化实现 [J]. 计算机系统应用. 2018, 27(4):151-156. (Song Dongfei Xu Hua. Parallel implementation of improved K-means algorithm based on Spark [J]. Computer System & Applications. 2018, 27(4): 151-156.)

[9] Arthur D,Vassilvitskii S. K-means+: the advantages of careful seeding [J]. SODA. 2007, 1027–1035.

[10] Zaharia M, Chowdhury M, Franklin M J, et al, Spark: clustercomputing with working sets [J]. HotCloud, 2010, 1-6.

[11] Wang Fei, Hugo H, Penya F. An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity [J] , Machine Learning and Data Mining in Pattern Recognition, 2017, 10358: 291-305.

[12] Li Zhong, Yuan Jinsha, Zhang Weihua. Fuzzy C-mean algorithm with morphology similarity distance [J]. Fuzzy systems and knowledge discovery, 2009:90-94.